Defence Science Journal, Vol. 74, No. 3, May 2024, pp. 389-398, DOI : 10.14429/dsj.74.18864

Military Decision Support with Actor and Critic Reinforcement Learning Agents

Jungmok Ma

Department of National Defense Science, Korea National Defense University, Nonsan, Republic of Korea
E-mail: jxm1023@gmail.com

ABSTRACT

While the recent advanced military operational concept requires intelligent support of command and control, Reinforcement Learning (RL) has not been actively studied in the military domain. This study points out the limitations of RL for military applications from a literature review and aims to improve the understanding of RL for military decision support under these limitations. Most of all, the black box characteristic of Deep RL makes the internal process difficult to understand, in addition to the complex simulation tools. A scalable weapon selection RL framework is built, which can be solved either by a tabular form or a neural network form. The transition of the Deep Q-Network (DQN) solution to the tabular form allows for effective comparison of the results to the Q-learning solution. Furthermore, rather than using one or two RL models selectively as before, RL models are divided into an actor and a critic, and systematically compared. A random agent, Q-learning and DQN agents as critics, a Policy Gradient (PG) agent as an actor, Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO) agents as an actor-critic approach are designed, trained, and tested. The performance results show that the trained DQN and PPO agents are the best decision support candidates for the weapon selection RL framework.

Keywords: Reinforcement learning; Military decision support; Actor and critic; Weapon selection; Battle damage assessment

NOMENCLATURE

RL	: Reinforcement learning
DRL	: Deep reinforcement learning
t	: Time steps, t=0,1,2, ...
S	: A set of states
A	: A set of actions
P	: A transition probability
R	: Reward
γ	: Discount factor
G	: Expected cumulative reward
π	: Policy
$\nu()$	: Value function
$q()$	: State-value function
α	: Learning rate
w	: Weights
θ	: Policy's parameter vector
DQN	: Deep Q-networks
PG	: Policy gradient
PPO	: Proximal policy optimization
TRPO	: Trust region policy optimization
BDA	: Battle damage assessment

1. INTRODUCTION

In recent years, Multi-Domain Operations (MDO) has become a critical operational concept since the multiple domains can be threatened by adversarial states or non-state actors, and the holistic control of the domains is essential for military victory. Under MDO, a higher-level Command and Control (C2) is required to coordinate various resources across land, air, maritime, space, and cyberspace, and Artificial Intelligence (AI) has been recognized as the core component to support and guide the C2 in the process of OODA (Observe, Orient, Decide, Act) Loop[1,2].

AI techniques (or machine learning) are generally categorized as supervised, unsupervised, and Reinforcement Learning (RL). Supervised learning maps labeled data to labeled categories (classification) or known outputs (regression). Unsupervised learning discovers a pattern from unlabeled data. Both categories have shown various potential military applications such as image, speech and pattern recognition, cybersecurity and threat intelligence, military personnel management, etc.[3]

Unlike supervised and unsupervised learning, RL is dynamic in nature, so it has the benefit of dealing with complex and dynamic environments such as multi-domains. The well-known application of RL is a strategic game such as ATARI game agent[4], AlphaGo[5], and AlphaStar[6]. Recently, researchers tried to apply RL to military wargames[7-9] and strategic maneuver scenarios[2]. However, RL has some limitations for military applications. First, RL was combined with deep learning (deep RL or DRL) to deal with a complex input and output structure. As a characteristic of deep learning, DRL is a black box model. Moreover, simulation environments such as AFSIM (Advanced Framework for Simulation, Integration, and Modeling)[7] and OpSim[10] in military applications can add complexity and make it difficult to understand the results. Second, while

Received : 09 February 2023, Revised : 16 February 2023
Accepted : 23 January 2024, Online published : 10 May 2024

RL can solve diverse problems, it can be slow and require lots of training data[11]. Third, though there are many RL algorithms, researchers have applied one or two of them selectively and subjectively.

With these challenges, this study aims at improving the understanding of RL for military decision support. A scalable weapon selection RL framework will be built, which is a simulation model and can be solved either by a tabular form or a neural network form. The tabular form can help understand the neural network form. Also, rather than using one or two RL models selectively, RL models will be divided as actors and critics, and systematically compared. Finally, important studies of RL for military decision support will be reviewed.

2. REINFORCEMENT LEARNING AND MILITARY APPLICATIONS

2.1 Key Concepts of RL

RL is learning "how to map situations to actions" through trial and error in order to maximize the cumulative reward[12]. In other words, a learning agent or decision maker should observe states (S_t) to gather information while interacting with its environment and take actions (A_t) to achieve a goal or maximum cumulative reward at each time steps, t=0, 1, 2, ...

RL can be formalized using the Markov Decision Process (MDP). MDP consists of a 5-tuple (S, A, P, R, γ) where, S is a set of states with a current state $s \epsilon S$, A is a set of actions, $P_a(s' \mid s)$ is a transition probability from the state s to a new state s' by selecting an action $a \epsilon A$, $R(s, s')$ is a reward for the action, and $\gamma \epsilon [0,1]$ is a discount factor that adjusts the weights of rewards in the future[13].

The expected cumulative reward or return with the discount factor can be defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (1)$$

A policy π that an agent follows is a mapping function from states to probabilities of choosing actions, which is defined as

$$\pi(a \mid s) = P[A_t = a \mid S_t = s] \quad (2)$$

Then, the value function $v_\pi(s)$ and the state-value function $q_\pi(s,a)$ under a policy π can be defined as the expected returns

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] \quad (3)$$

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a] \quad (4)$$

The optimal policy π_* gives the highest value function $v_\pi(s)$ or the highest state-value function $q_\pi(s,a)$.

There are a couple of ways to classify RL algorithms. The first one is model-based and model-free. For the model-free method, a transition probability P and a reward R in the 5-tuple of MDP are not given to an agent. This paper only deals with the model-free method. The second one is value-based and policy-based. While valued-based methods (critic agents) construct a value function and derive the optimal policy implicitly, policy-based methods (actor agents) find the optimal policy function directly that maps states to actions. Actor-critic methods attempt to use both value-based and policy-based methods. The third one is on-policy and off-policy. On-policy methods learn a policy from the data generated by the agent's actions, whereas off-policy methods learn a policy separately from the agent's actions.

2.2 RL in Military Applications

In this section, RL studies for military decision support are reviewed. Military decision support is defined as assisting the combat and mission planning from controlling a single asset (physical asset (solider, weapon, unmanned system, etc.) and non-physical asset (software, network, frequency band, etc.)) to coordinating multiple assets for achieving the military objective.

Unmanned Aerial Vehicles (UAVs) are the most popularly used unmanned systems for both civilian and military purposes. There were many RL studies for UAVs, even though they were not for direct military applications[14]. The studies could be divided into two areas: control and path planning/navigation. The RL studies of UAV control included longitudinal and lateral control[15], attitude control[16], and swarm control[17]. Path planning/navigation consisted of finding an optimal path[18,19] and avoiding obstacles[20-23]. RL was used for not only UAVs but also other unmanned platforms such as Unmanned Ground Vehicles (UGVs)[24,25] and Unmanned Underwater Vehicles (UUVs)[26].

Furthermore, RL was popularly used for cybersecurity[27,28] in an adversarial environment. UAV networks[29,30], home networks[31], and wideband controller[32] were used as RL agents for intrusion detection and prevention. A gateway[33], attacker[34], mobile device[35], underwater sensor[36] were used as RL agents for identity and access management.

For more military-specific studies, Yan*, et al.* (2020) proposed Deep Q-Networks (DQN)-based UAV path planning in dynamic environments with potential threats[37]. An UAV survival probability was considered under radar detection and missile attack, which was implemented on the STAGE simulation tool. You*, et al.* (2019) proposed to train the maneuvering strategies of Unmanned Combat Air Vehicles (UCAVs) using the Deep Deterministic Policy Gradient (DDPG)[38]. A 3-D space game environment was built to simulate a UCAV and an electronic attacker with radars. Wang*, et al.* (2020) developed a middleware based on commercial air combat simulation software and trained a fighter aircraft for the maneuvering strategy of 1 vs. 1 air combat[39]. DQN was used for the aircraft RL agent in the proposed alternate freeze game, which allows that one agent to act while the other is frozen for a better understanding of the learning process.

Zhang*, et al.* (2020) investigated the potential of AI-assisted planning for the US defence community[7]. As a simulation environment, OpenAI's Gym RL framework and AFSIM were integrated as a low-fidelity version of AFSIM. Two problem scenarios were formulated as follows: The first scenario was 1-D with three components: Blue fighter, Blue jammer, and Red Surface-to-Air Missile (SAM). The goal was to learn what times and distances to deploy the fighter and jammer to destroy the SAM, i.e., go or stop the problem. A planner agent was trained using Generative Adversarial Networks (GAN) and Q-learning. The second scenario was 2-D with three components: Blue UAVs, Red SAM, and Red air target that could attack the Blue UAVs. A planner agent was

trained using Asynchronous Advantage Actor-Critic (A3C) and Proximal Policy Optimization (PPO). Soleyman and Khosla (2020) proposed multi-agent mission planning in the AFSIM environment[8]. Up to 6 vs. 6 Blue and Red fighters could be simulated and trained using A3C in their framework.

Boron and Darken (2020) tried to validate the tactical principles of mass and economy using RL[9]. For a simulation environment, a discretized 10-by-10 map grid of squares was created with deterministic and stochastic Lanchester's modern combat equations. Platoon entities conducting offensive operations were trained using Vanilla Policy Gradient (VPG), PPO, and Trust Region Policy Optimization (TRPO) in three different force configurations: 2 vs. 1, 2 vs. 2, 3 vs. 2 scenarios. Fu*, et al.* (2020) built a digital battlefield for air-to-ground operations using Unreal Engine 4, and a neural network (Alpha C2) was trained using PPO[40]. Two training scenarios were designed, and the performance was compared with Expert C2 (an expert system designed by known rules).

Goecks*, et al.* (2021) focused on the connection between games and military applications[10]. First, the StarCraft II Learning Environment was used as an environment to implement a brigade-scale offensive scenario. StarCraft II units were mapped to military units and trained using A3C. Second, the OpSim simulator with the OpenAI Gym was used to implement the same offensive scenario. The military units were trained by Advantage Actor-Critic (A2C) and compared with an expert system designed by military doctrinal rules. Zhang*, et al.* (2022) built a multiple domain cyberspace attack and defence game with two agents using Python, which aimed at maximizing the defender's total reward[41]. The defender was trained using DQN, DDPG, and DDPG with reward randomization, and the performance was compared.

One distinct sub-problem of military planning is weapon target assignment (WTA). WTA is a process of evaluating threats or targets and allocating weapons. The WTA problem was solved by traditional optimization techniques and intelligent algorithms such as game theory, genetic algorithm, particle swarm algorithm, etc.[42] Wu*, et al.* (2022) built the problem of multi-warhead penetration and striking multi-target using Python[42]. The defence strategy was trained using Soft Actor-Critic (SAC) and compared with the optimal rule model. Mouton*, et al.* (2011) built a simple grid world to simulate an air defence operator who needed to assign weapon systems to engage enemy aircraft[43]. The defence operator was trained using the temporal-difference algorithm and Q-learning, and their results were compared. Shin*, et al.* (2020) introduced the WTA problem with inference constraints, and DQN was used to train the assignment strategy[44]. In order to reduce the computational complexity of multi-agent RL, mean field Q-learning was applied, and the result was compared with traditional mixed integer linear programming.

3. SIMULATION MODEL AS ENVIRONMENT

Provide As a learning environment, a simulation model was built that requires a series of weapon selections with Battle Damage Assessment (BDA) when a target is given. The target can be acquired by manned and unmanned systems with its required damage between 10~50, as shown in Table 1. There are seven weapons, from Weapon1 to Weapon7, and each weapon has its own damage profile or damage distribution. For example, Weapon1 shows the discrete distribution of damage depending on the battlefield condition. Once one weapon

Table 1. Weapon selection RL framework

Required damage		**Integer between 10 and 50**
Damage distribution	Weapon1	1 1 1 1 2 2
	Weapon2	3 3 3 3 4 5
	Weapon3	4 4 4 6 6 7
	Weapon4	8 8 8 9 9 10
	Weapon5	10 10 10 20 20 20
	Weapon6	15 15 15 25 25 30
	Weapon7	30 30 30 40 40 50

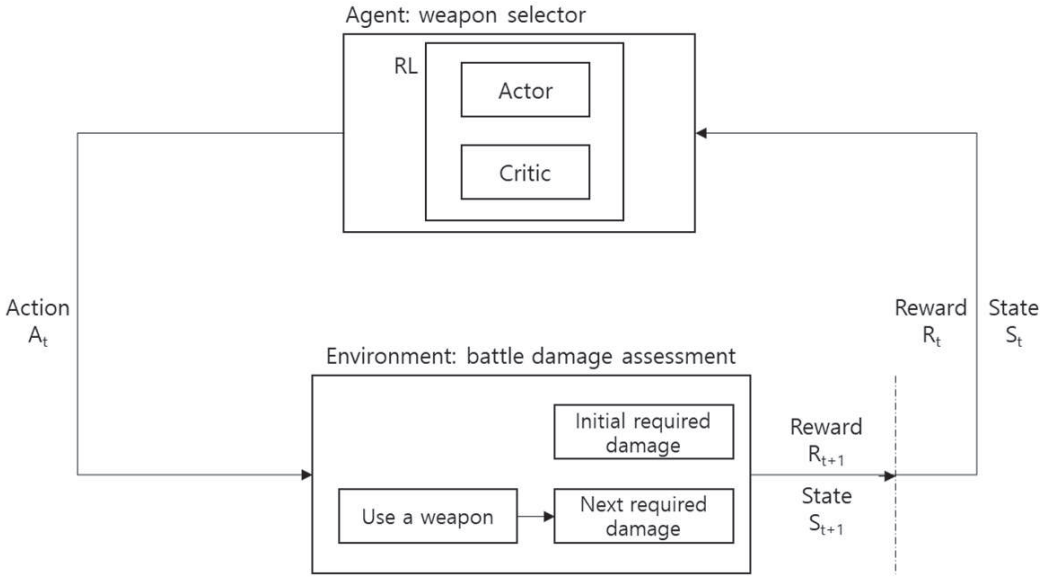

Figure 1. Weapon selection RL framework.

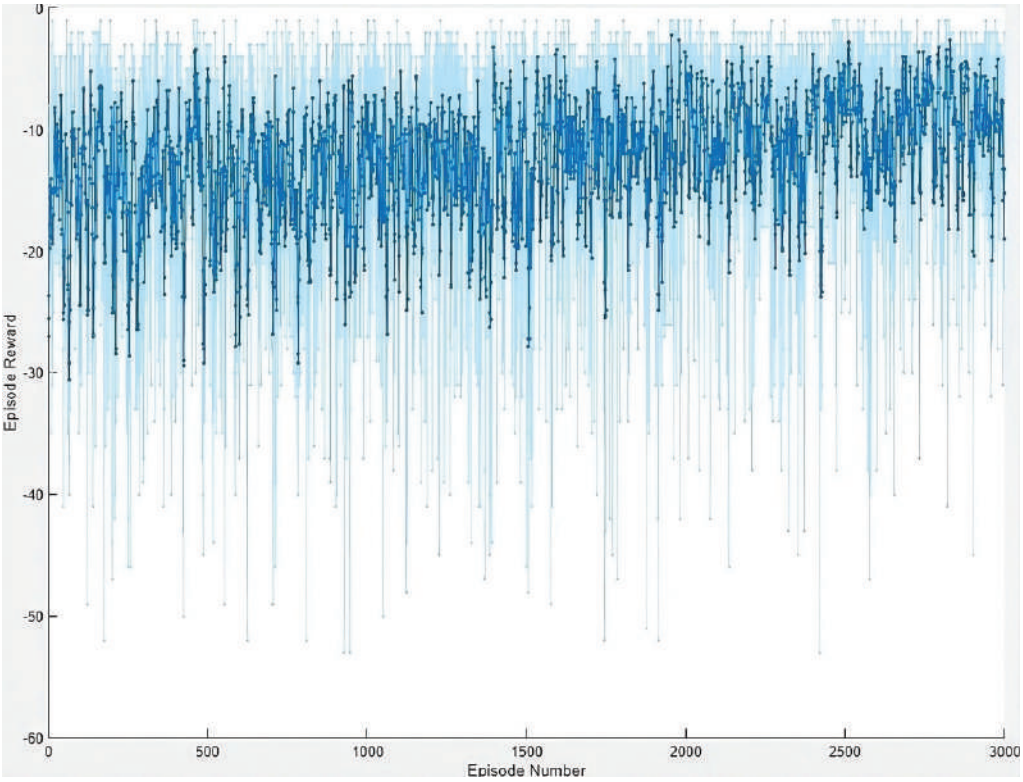

Figure 2. Episode number and reward for Q-learning agent training.

is selected and used, the BDA is conducted by manned and unmanned systems. Then, the next required damage for the target is provided, which is the difference between the required damage and the damage caused by the selected weapon. The selection continues until the required damage is achieved, and the weapon selection RL framework is depicted in Fig. 1.

The states represent the required damages from 0 to 50. So, this simulation model has discrete action and observation spaces. The simulation will end if the next required damage is less than or equal to 0. The goal of this simulation is to achieve the required damage quickly. Therefore, whenever a weapon is selected, there will be a penalty of -1. Also, if the next required damage is negative (more damage occurrs than the next required damage), the cumulative reward will be the cumulative penalties from weapon selections plus excess damage.

The weapon selection RL framework was built using Simulink (R2021b version) in MATLAB, which is a block diagram-based simulation tool. The constructed RL environment was verified using various cases. For example, one episode (a series of steps until the simulation ends) showed that the initial required damage was 32, and Weapon2 was selected with a penalty of -1. The damage was 3, and the next required damage was 29. Then, Weapon1 was selected with the damage of 1, and Weapon7 was selected with the damage of 40. Since the next required damage was negative (-12), the simulation ended with a return of -15 (-3 for weapon selections and -12 from excess damage).

4. AGENT MODELLING AND TRAINING

In this section, an agent that interacts with the BDA environment in Fig. 1 will be modelled and trained. In order to easily understand the built agents, a tabular form solution will be first explained with a Q-learning agent. And then, DRL agents and advanced actor-critic approaches will be used. All the agents will be realized using MATLAB (R2021b version). Note that the weapon selection RL framework has discrete action and observation spaces, so continuous space algorithms such as DDPG, TD3 (Twin Delayed DDPG), and SAC in Section 2.2 cannot be used.

4.1 Q-learning Agent as a Critic and Random Agent

Q-learning is defined as[45]

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (5)$$

where, Q is the action-value function to approximate the optimal action-state function q_*; α is the learning rate; γ is the discount factor. The Q-learning algorithm begins with the initialization of $Q(s, a)$ or Q values and then iterates as follows: choosing A with S using Q values (either exploration or exploitation), taking action A and observing R and S', updating $Q(s, a)$ from Eqn. (5). To balance exploration and exploitation, the ε-greedy method can be adopted, which selects a random action with probability $\varepsilon\epsilon[0,1]$ and otherwise the action with maximum Q value. The Q-learning agent was created using rlQ Agent() and rlQ Value Representation() functions in MATLAB.

When all the Q values were set to be 0 (7×51 table), the Q-learning agent only chose Weapon1, regardless of the required damage. The random agent was realized as a random number of Q values for each episode. Therefore, no training is required for the random agent. In order to determine the number of episodes for testing other agents, the performance test for the random agent was conducted first. The random

agent interacted with the weapon selection RL framework in Section 3 and resulted in an average return of -15.1, -15.23, -15.34 for randomly generated 1k, 3k, and 5k episodes. Since there are no big differences among them, 3k episodes were selected for the performance test of other agents.

The Q-learning agent was trained with a learning rate of 0.005 and a discount factor of 0.99. Figure 2 shows the episode numbers and rewards, and the average returns during training are summarized in Table 2. Due to the uncertainty of the simulation model (initial required damage and damage distribution), the episode reward shows fluctuations, and after 3k episodes, the average return reaches -13.2. Also, due to the randomness, increasing the episode number does not ensure a better average return (1k vs. 2k and 3k).

Table 2. Average return during training

Episode number	500	1k	2k	3k
Average return during training	-17	-8	-18.4	-13.2

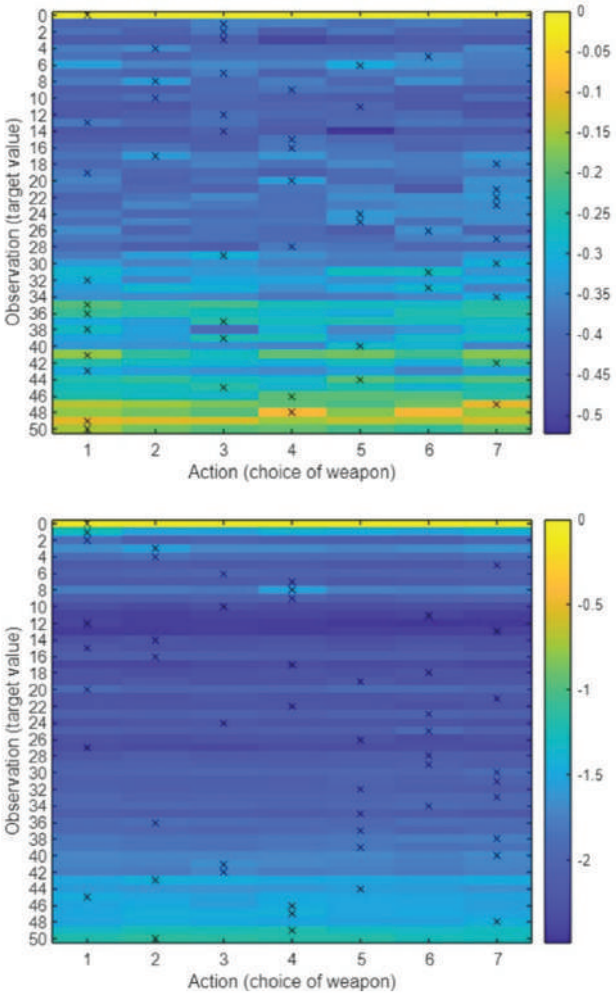

Figure 3. Q table with learned action '×' after 500 (left) and 3k (right) episodes.

Figure 3 shows the tabular form of the Q-learning agent after 500 and 3k episodes. The × mark represents the optimal action or highest Q value action in the given state. Since the weapon damage distributions are intentionally simplified, the right half of a dome shape should appear as good actions. For example, when the required damage (target value) is close to 1, Weapon1 should be selected, and when close to 50, Weapon7 should be used. It can be observed that after 3k episodes, better actions can be selected in comparison with 500 episodes, but still there is a space to be improved.

4.2 DQN Agent as a Critic

DQN was initially proposed to adopt a neural network in order to learn control policies from video data in complex environments, e.g., Atari 2600 games, as a variant of Q-learning[4]. Q-network is a neural network function approximator with weights w. When a certain number, C, of updates are conducted, another network takes the weights and fixes the weights for the next C updates of w. The outputs of the duplicate network are used as the Q-learning targets for the next C updates of w. The update rule is defined by[4]

$$w_{t+1} = w_t + \alpha[R_{t+1} + \gamma \max_a \tilde{q}(S_{t+1}, a, w_t) - \hat{q}(S_t, A_t, w_t)]\nabla q(S_t, A_t, w_t) \quad (6)$$

where, $\hat{q}$ and $\tilde{q}$ are the outputs of Q-network and the duplicate network. Minih, *et al.* (2013) also utilized the experience replay, which stores experience in a replay memory and provides samples for updates[4].

The DQN agent was created using rlDQNAgent() and rlQValueRepresentation() functions in MATLAB. The structure of the Q-network was designed as two paths of layers of inputs (actions and observations), and each path consisted of a scalar input layer, two hidden layers (50 neurons) with two ReLU activation function layers, one fully connected layer (50 neurons). The two paths were combined and then one ReLU activation layer and a single output layer were followed.

The DQN agent was trained with a learning rate of 0.001 and a discount rate of 0.99. Figure 4 shows the episode numbers and rewards. The average return during training is -3.4 after 3k episodes, which is a big improvement in comparison with the Q-learning agent.

Using the trained DQN agent with 3k episodes, a tabular form was built as shown in Fig. 5. Note that the trained DQN agent is not a tabular form but a neural network form. In comparison with the Q-learning agent in Fig. 3, it can be observed that the actions are much improved.

4.3 PG Agent as an Actor

The Policy Gradient (PG) method only takes states as an input and provides the probability of actions with the policy's parameter vector θ as[12].

$$\pi(a \mid s, \theta) = P[A_t = a \mid S_t = s, \theta_t = \theta] \quad (7)$$

The benefit of policy parameterization is that the action probability changes more smoothly than ε-greedy, and strong convergence is guaranteed than action-value methods.

The Monte-Carlo PG algorithm, REINFORCE, starts with the initialization of policy parameter θ and generates episodes $S_0, A_0, R_1, \ldots, A_{T-1}, R_T, S_T$ from $\pi(\cdot \mid \cdot, \theta)$. Then, updates for each step t=0,1, …, T-1[12].

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \quad (8)$$

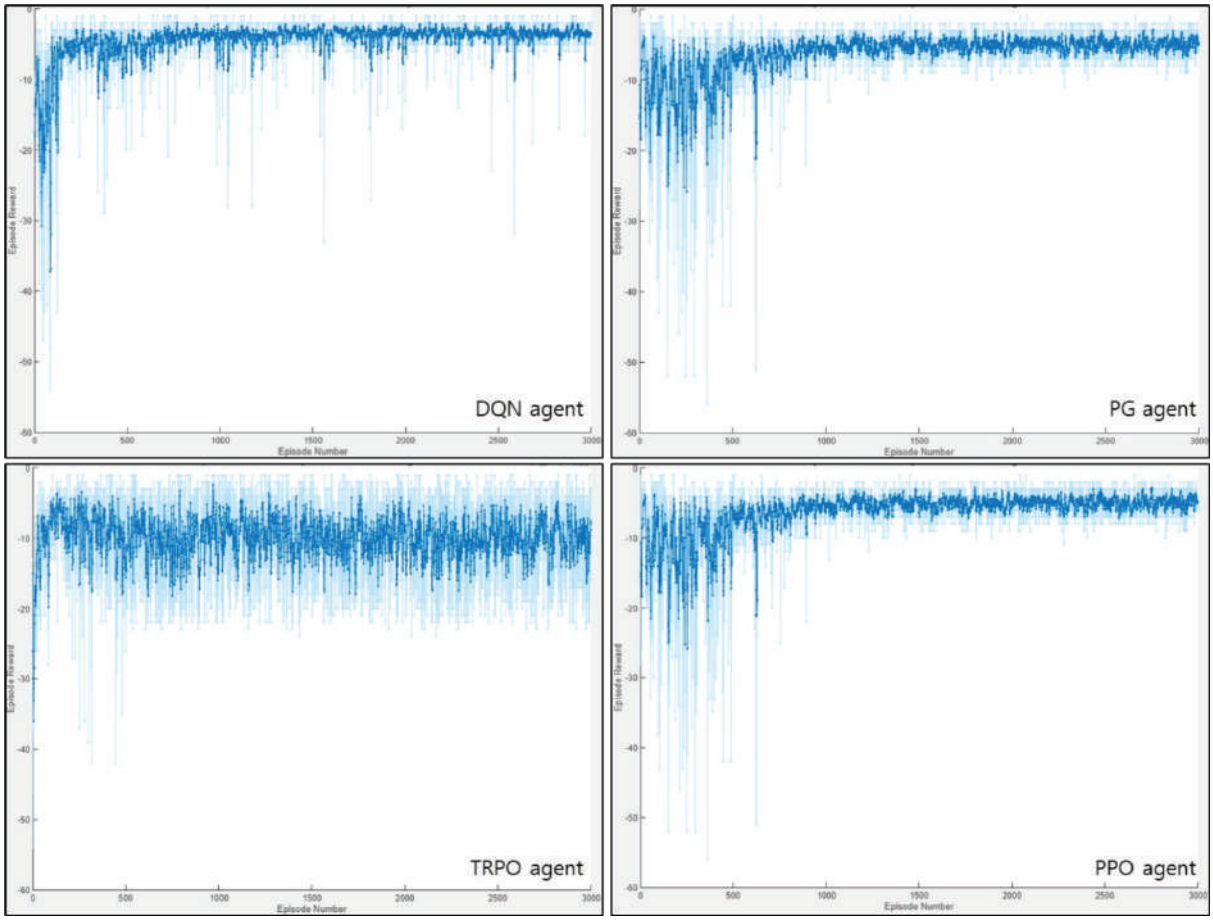

Figure 4. Episode number and reward for DQN, PG, TRPO, PPO agent during training.

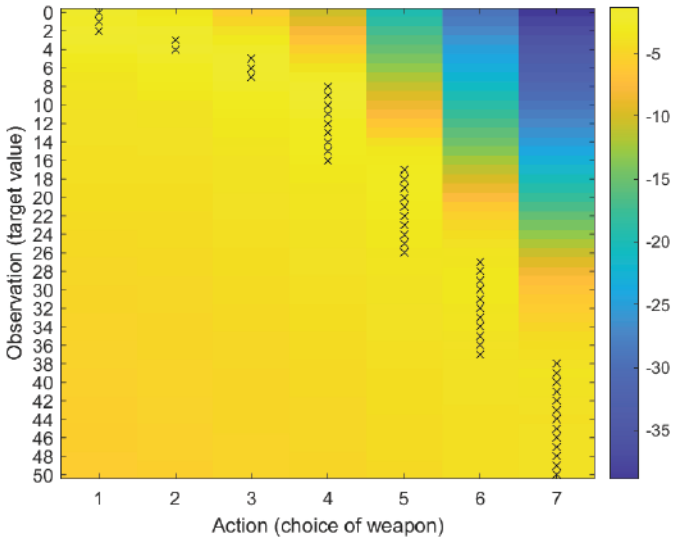

Figure 5. Transition of DQN agent to a tabular form.

$$\theta \leftarrow \theta + \alpha \gamma^{t} G \nabla \ln \pi(A_t \mid S_t, \theta) \tag{9}$$

The PG agent was created using rlPG Agent() and rl Stochastic Actor Representation() functions in MATLAB. The structure of the PG actor network was designed as a scalar input layer for observations, two hidden layers (50 neurons) with two ReLU activation function layers, one fully connected layer (7 neurons for 7 weapons), and a Softmax function layer. The PG agent was trained with a learning rate of 0.001 and a discount rate of 0.99. Figure 4 shows the episode numbers and rewards. The average return during training is -10 after 3k episodes.

4.4 Advanced Actor-Critic Approaches: TRPO and PPO Agent

A Q value critic (Q-learning and DQN) and a PG actor were used as separate RL agents as shown in Fig. 6. The actor-critic approach uses both actor and critic agents, not just one of them. Fig. 6 graphically shows the difference between them, and the actor-critic approach can be described as follows: After observing states in the environment, actions and Q values are generated with an observed reward. The action leads to a new state, and the critic produces a new value. These interactions give the original and updated values of the original state, and the actor-critic approach algorithm trains the actor and the critic with the information.

TRPO[46] was proposed to guarantee monotonic improvement from standard PG methods. The Kullback-Leibler (KL) divergence between the old and current polies was used as a trust region constraint, and the trust region can control the difference between the old and current polies. Then, the constrained optimization problem could be formulated to find the best policy in order to maximize the expected discounted reward. To solve this problem practically, the surrogate objective function was used using Monte Carlo simulation.

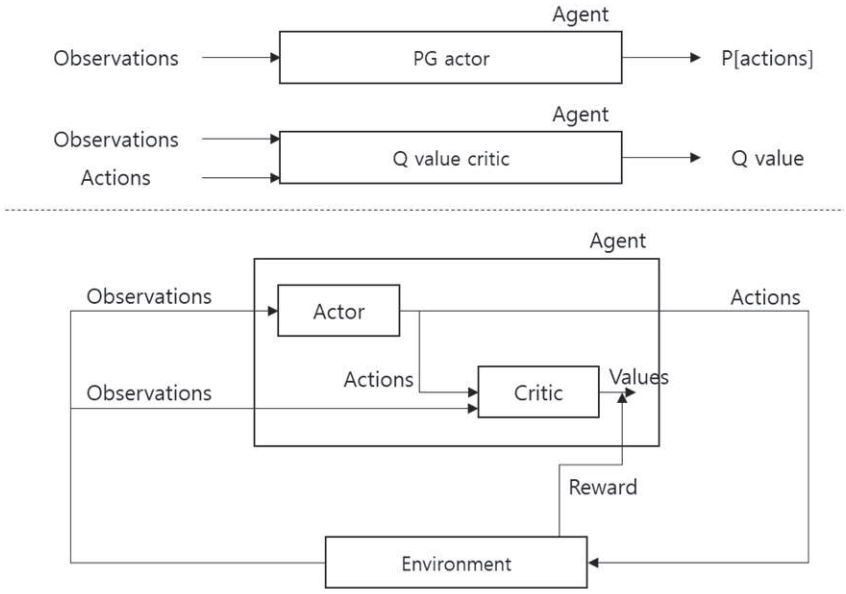

Figure 6. Individual actor and critic (top) and actor-critic approach (bottom).

PPO[47] was proposed as a simplified version of TRPO with faster and more efficient characteristics. The surrogate objective function in TRPO was modified using the clipping of a constraint, which made TRPO a first-order algorithm.

The TRPO agent was created using rlTRPO Agent(),rl Stochastic Actor Representation(), rlQ Value Representation() functions and the PPO agent was created using rlPPO Agent(),rl Stochastic Acto r Representation ()and rlQ Value Representation() functions in MATLAB. As an actor network for both TRPO and PPO, similar to the PG actor network, a scalar input layer for observations, two hidden layers (50 neurons) with two ReLU activation function layers, one fully connected layer (7 neurons for 7 weapons), and a Softmax function layer were organized as a network. As a critic network for both TRPO and PPO, a scalar input layer, two hidden layers (50 neurons) with two ReLU activation function layers, and one fully connected layer (50 neurons) were organized as a network. Both TRPO and PPO agents were trained with a learning rate of 0.001 and a discount rate of 0.99. Fig. 4 shows the episode numbers and rewards. The average return during training is -7.6 for the TRPO agent and -4.8 for the PPO agent after 3k episodes.

Table 3. Average return for RL agents trained using 3k episodes

	Critic			Actor	Actor-Critic	
	Random	**Q-learning**	**DQN**	**PG**	**TRPO**	**PPO**
Avg. return for eval.	-15.2	-8.4	-3.4	-10.4	-10.2	-4.9

4.5 Performance Test of Agents

The performance test of the trained agents was conducted with randomly generated 3k episodes in the simulation model. The random agent's average return was about -15, and all other RL agents showed better results, as shown in Table 3. Among them, the trained DQN and PPO agents showed the best performance.

In order to see the effect of training episode numbers, when the RL agents were trained using 5k episodes and tested using 3k episodes, the resulting ranking stayed the same as in Table 3. The histograms of rewards for the RL agents are presented in Fig. 7. Since the X-Axis is the absolute value of rewards, smaller is better, and the trained DQN and PPO agents show smaller means and deviations than the other trained agents.

In order to check the further improvement in performance from more training, only DQN and PPO agents were trained using 10k episodes and tested using 3k episodes, as shown in Table 4. However, it was found that the performance was not improved from Table 3. In conclusion, the trained DQN and PPO agents show the best overall performance and can be used as the weapon selection decision support system.

Table 4. Average return for RL agents trained using 10k episodes

	DQN	**PPO**
Average return during training (10k episodes)	-3.2	-5
Average return for evaluation (3k episodes)	-3.42	-4.95

5. CONCLUSION

This study pointed out the limitations of RL for military applications from a literature review and aimed at improving the understanding of RL for military decision support under the limitations. Most of all, the black box characteristic of Deep RL makes things difficult to understand, in addition to the complex simulation tools. A scalable weapon selection RL framework was built, which can be solved either by a tabular form or a neural network form. The transition of the DQN solution to the tabular form made it easier to compare the result to the Q-learning solution. Furthermore, rather than using one or two RL models selectively as before, RL models were divided into an actor and a critic and systematically compared. A random agent based on Q-learning, Q-learning and DQN agents as a

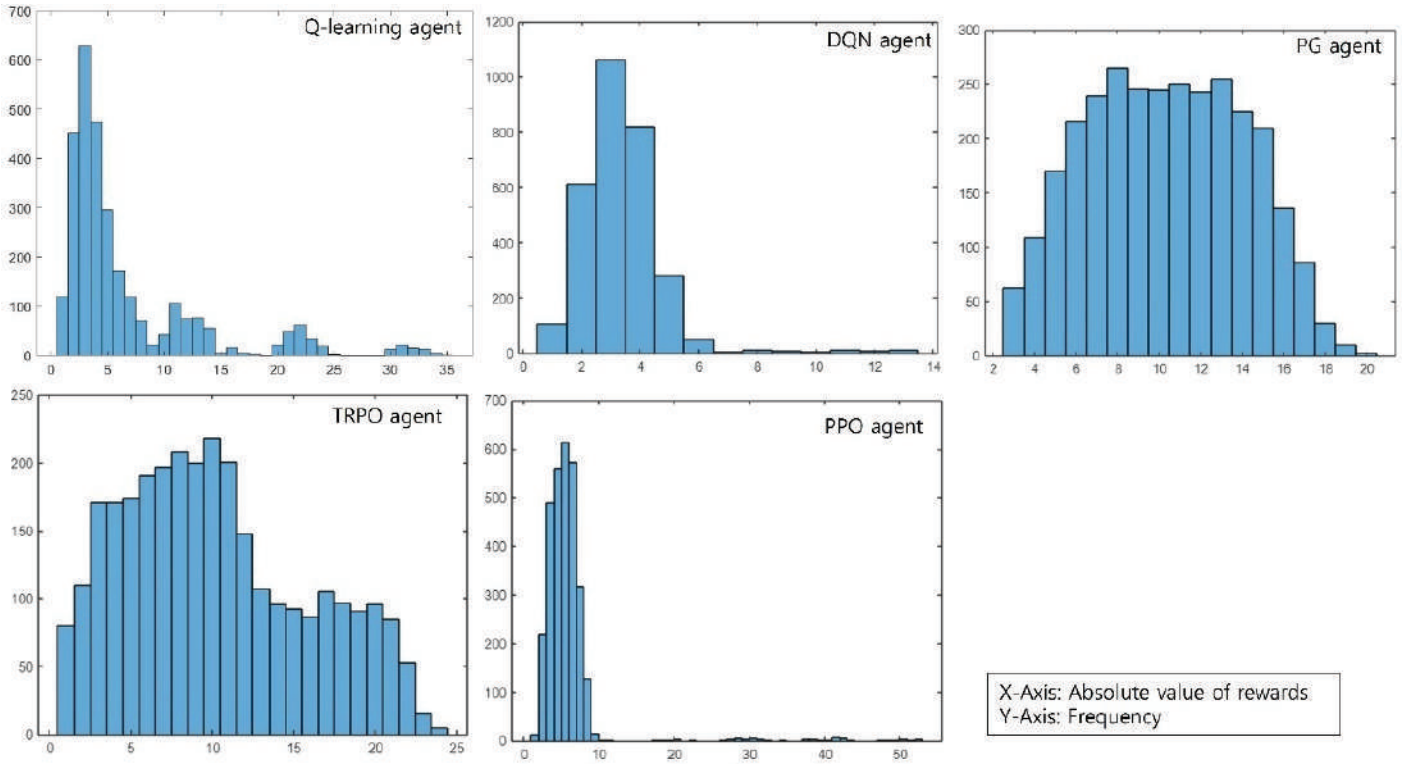

Figure 7. Histogram of rewards for RL agents trained using 5k episodes.

critic, a PG agent as an actor, TRPO and PPO agents as an actor-critic approach were designed, trained, and tested. The performance results showed that DQN and PPO agents were the best decision support candidates for the weapon selection RL framework.

The limitations of this study are as follows: First, the proposed scalable weapon selection RL framework required a series of decisions within the same selection problem. It can be extended to a decision problem with a complex hierarchy. Second, in order to improve the understanding of DRL, the tabular form of solutions was used. Other effective, explainable methods can be further explored. Third, the simulation model had a discrete space, soRL models with a continuous space could not be used. Even though this study has these limitations, it is very important to bridge the gap between researchers and practitioners, and RL models need to be studied more actively for a military decision support system. It is not for boosting the advent of killer robots but for peaceful usage of AI technologies.

REFERENCES

1. Asher, D.E.; Basak, A.; Fernandez R..; Sharma, P.K.; Zaroukian, E.G.*, et al.* Strategic maneuver and disruption with reinforcement learning approaches for multi-agent coordination. *J. Def. Model. Simul.,* 2023, **20(4)**, 509-526. doi:10.1177/15485129221104096
2. Fernandez, R.; Asher, D.E.; Basak, A.; Sharma, P.; Zaroukian, E.G.*, et al.*Multi-Agent Coordination for strategic maneuver with a survey of reinforcement learning. DEVCOM Army Research Laboratory, 2021, 1-33. https://apps.dtic.mil/sti/pdfs/AD1154872.pdf [Accessed on 1 February 2023]
3. Galán, J.J.; Carrasco, R.A. & LaTorre, A.military applications of machine learning: A bibliometric perspective. *Mathematics*, 2022, **10(9)**, 1397. doi: 10.3390/math10091397
4. Mnih, V.; Kavukcuoglu, D.; Silver, D.; Graves, A.; Antonoglou, I.*, et al.* Playing atari with deep reinforcement learning. arXiv:1312.5602, 2013.
5. Silver, D.; Huang, A.; Maddison, C; Guez, A.; Sifre, L.*, et al.*Mastering the game of Go with deep neural networks and tree search. *Nature*,2016,**529**, 484–489. doi: 10.1038/nature16961
6. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.*, et al.*Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 2019,**575**, 350–354. doi: 10.1038/s41586-019-1724-z
7. Zhang, L.A.; Xu, J.; Gold, D.; Hagen, J.; Kochhar, A.K.*; et al.* Air dominance through machine learning: A preliminary exploration of artificial intelligence–Assisted Mission Planning. RAND Corporation, 2020. doi: 10.7249/RR4311
8. Soleyman, S. & Khosla, D. Multi-agent mission planning with reinforcement learning. *In* Proceedings of AAAI Symposium on the 2nd Workshop on Deep Models and Artificial Intelligence for Defense Applications: Potentials, Theories, Practices, Tools, and Risks, November 11-12, Virtual, 2020. https://ceur-ws.org/Vol-2819/session1paper1.pdf [Accessed on 1 February 2023]
9. Boron, J. & Darken,C. Developing combat behavior through reinforcement learning in war games and simulations. *In* Proceedings of2020 IEEE Conference on

Games (CoG), Osaka, Japan, 2020, 728-731.
doi: 10.1109/CoG47356.2020.9231609

10. Goecks, V.G.; Waytowich, N.; Asher, D.E.; Park, S.J.; Mittrick, M.*, et al.*On games and simulators as a platform for development of artificial intelligence for command and control. *J. Def. Model. Simul.*, 2022,**20(4)**, 495-508. doi:10.1177/15485129221083278
11. Sharma, R.; Prateek, M. &Sinha, A.K.Use of reinforcement learning as a challenge: A review. *Int. J. Comput. Appl.*, 2013, **69(22)**, 28-34.
doi: 10.5120/12105-8332
12. Sutton, R.S. & Barto, A.G. Reinforcement learning: An introduction. Second Edition, MIT Press, Cambridge, MA, 2018.
13. Pröllochs, N. & Feuerriegel, S. Reinforcement Learning in R. arXiv:1810.00240v1, 2018.
14. Azar, A.T.; Koubaa, A.;Mohamed, N.A.; Ibrahim, H.A.; Ibrahim, Z.F.*, et al.* Drone deep reinforcement learning: A review. *Electronics*, 2021, **10(9)**, 999.
doi: 10.3390/electronics10090999
15. Bøhn, E.; Coates, E.M.; Moe, S. & Johansen, T.A.Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14, 2019, 523–533.
16. Koch, W.; Mancuso, R.; West, R. & Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans.Cyber Phys. Syst.*, 2019,**3**, 1–21.
doi: 10.1145/3301273
17. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.& Piao, C.Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE J. Sel. Areas Commun.*, 2018, **8(36)**, 2059–2070.
doi: 10.1109/JSAC.2018.2864373
18. Qu, C.; Gai, W.; Zhong, M. & Zhang, J. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. *Appl. Soft Comput.*, 2020, **89**, 106099.
doi: 10.1016/j.asoc.2020.106099
19. Bayerlein, H.; Theile, M.; Caccamo, M.& Gesbert, D.UAV path planning for wireless data harvesting: A deep reinforcement learning approach. arXiv:2007.00544, 2020.
20. Jiang, S.; Jiang, C.& Jiang, W.Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools. *ISPRS J. Photogramm. Remote Sens.*, 2020, **167**, 230–251.
doi: 10.1016/j.isprsjprs.2020.04.016
21. He, L.; Aouf, N.; Whidborne, J.F. & Song, B.Deep reinforcement learning based local planner for UAV obstacle avoidance using demonstration data. arXiv:2008.02521, 2020.
22. Singla, A.; Padakandla, S.& Bhatnagar, S.Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge. *IEEE Trans. Intell. Transp. Syst.*, 2021, **22(1)**, 107-118.
doi: 10.1109/TITS.2019.2954952.
23. Shin, S.Y.; Kang, Y.W. & Kim, Y.G. Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot. *Appl. Sci.*, 2019, **9(24)**, 5571.
doi: 10.3390/app9245571.
24. Wei, M.; Wang, S.; Zheng, J.& Chen,D. UGV navigation optimization aided by reinforcement learning-based path tracking. IEEE Access, 2018,**6**, 57814-57825.
doi: 10.1109/ACCESS.2018.2872751
25. Sivashangaran, S. & Zheng, M. Intelligent autonomous navigation of car-like unmanned ground vehicle via deep reinforcement learning. *IFAC-Papers OnLine*, 2021,**54(20)**,218-225.
doi: 10.1016/j.ifacol.2021.11.178
26. Chu, Z.; Sun, B.; Zhu, D.; Zhang, M. & Luo, C. Motion control of unmanned underwater vehicles via deep imitation reinforcement learning algorithm. *IET Intell. Transp. Syst.*, 2020, **14**, 764-774.
doi: 10.1049/iet-its.2019.0273
27. Adawadkar, A.M.K. &Kulkarni, N. Cyber-security and reinforcement learning - A brief survey. *Eng. Appl. Artif. Intell.*, 2022,**114**,105116.
doi: 10.1016/j.engappai.2022.105116
28. Sewak, M.; Sahay, S. & Rathore, H. DRLDO A novel DRL based de obfuscation system for defence against metamorphic malware. *Def. Sci. J.*, 2021,**71(1)**, 55-65.
doi: 10.14429/dsj.71.15780
29. Tao, J.; Han, T. &Li, R.Deep-Reinforcement-learning-based intrusion detection in aerial computing networks. *IEEE Network*, 2021,**35(4)**,66-72.
doi: 10.1109/MNET.011.2100068
30. Bouhamed,O.; Bouachir,O.; Aloqaily, M.& Ridhawi, I.A. Lightweight IDS for UAV networks: A periodic deep reinforcement learning-based approach. *In* Proceedings of 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 2021, 1032-1037.
31. Heartfield, R.; Loukas, G.; Bezemskij, A.& Panaousis, E. Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. *IEEE Trans. Inf. Forensics Secur.*, 2021, **16**, 1720-1735.
doi: 10.1109/TIFS.2020.3042049
32. Aref,M.A.; Jayaweera, S.K., & Machuzak, S. Multi-agent reinforcement learning based cognitive anti-jamming. *In* Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 2017, 1-6.
doi: 10.1109/WCNC.2017.7925694
33. Ferdowsi, A. & Saad, W. Deep learning for signal authentication and security in massive internet-of-things systems. *IEEE Trans. Wirel. Commun.*, 2019,**67(2)**, 1371-1387.
doi: 10.1109/TCOMM.2018.2878025
34. Gao, N.; Ni, Q.; Feng, D.; Jing, X. &Cao, Y. Physical layer authentication under intelligent spoofing in wireless sensor networks. *Signal Processing*, 2020,**166**, 107272.
doi: 10.1016/j.sigpro.2019.107272
35. Lu, X.; Xiao, L.; Xu, T.; Zhao, Y.; Tang, Y.*, et al.* Reinforcement learning based PHY authentication for

VANETs. *IEEE Trans. Veh. Technol.*, 2020,**69(3)**, 3068-3079.
doi: 10.1109/TVT.2020.2967026

36. Xiao, L.; Sheng, G.; Wan,X.; Su, W.& Cheng, P. Learning-based PHY-layer authentication for underwater sensor networks. *IEEE Commun. Lett.*, 2019,**23(1)**, 60-63.
doi: 10.1109/LCOMM.2018.2877317
37. Yan, C.; Xiang, X. & Wang, C. Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *J. Intell. Robot Syst.*, 2020,**98**, 297–309.
doi: 10.1007/s10846-019-01073-3
38. You,S.; Diao, M. &Gao, L. Deep reinforcement learning for target searching in cognitive electronic warfare. *IEEE Access*, 2019,**7**, 37432-37447.
doi: 10.1109/ACCESS.2019.2905649
39. Wang, Z.; Li, H.; Wu, H. &Wu, Z. Improving maneuver strategy in air combat by alternate freeze games with a deep reinforcement learning algorithm. *Math. Probl. Eng.*,2020,**2020**, 1-17.
doi: 10.1155/2020/7180639
40. Fu,Q.; Fan, C.L.; Song, Y. & Guo, X.K. Alpha C2–An intelligent air defense commander independent of human decision-making. *IEEE Access*, 2020,**8**,87504-87516.
doi: 10.1109/ACCESS.2020.2993459
41. Zhang, L.; Pan, Y.; Liu, Y.; Zheng, Q. &Pan, Z. Multiple domain cyberspace attack and defense game based on reward randomization reinforcement learning. arXiv:2205.10990, 2022.
42. Wu, Y.; Lei, Y.; Zhu,Z.; Yang, X. &Li, Q. Dynamic multi target assignment based on deep reinforcement learning. *IEEE Access*, 2022,**10**, 75998-76007.
doi: 10.1109/ACCESS.2022.3190972
43. Mouton,H.; Roodt, J. &Roux, H. Applying reinforcement learning to the weapon assignment problem in air defence. *S. Afr. j. Mil. Stud.*, 2011, **39(2)**, 123-140.
doi: 10.5787/39-2-115
44. Shin, M.K.; Park, S.S.; Lee, D.; & Choi, H.L. Mean field game based reinforcement learning for weapon-target assignment. *J. KIMS Technol.*, 2020, **23(4)**, 337-345.
doi: 10.9766/KIMST.2020.23.4.337
45. Watkins, C.J.C.H. & Dayan, P.Q-learning. Mach. Learn., 1992, **8**, 279-292.
doi: 10.1007/BF00992698
46. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I. & Abbeel, P. Trust Region Policy Optimization. arXiv:1502.05477, 2015.
47. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. &Klimov, O. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.

CONTRIBUTORS

Dr Jungmok Ma obtained his PhD in Industrial Engineering from University of Illinois at Urbana-Champaign. He is a Professor in the Department of National Defense Science, Korea National Defense University. His research interest includes data analytics and National defense modeling.